Titel: Labor 10 – Init Systeme

Klasse: 3BHIF

Name: Haiden

Gruppe: 01

Aufgabe: 18.03.2020 Abgabe: 01.04.2020

Inhaltsverzeichnis

[1 Init-Systeme Linux 1](#_Toc36580192)

[1.1 Welches System (systemd oder init.d) verwendet dein Linux? 1](#_Toc36580193)

[1.2 Gibt es im Verzeichnis /etc/init.d Dateien? Wozu? 1](#_Toc36580194)

[1.3 Welche units gibt es auf deinem System? 1](#_Toc36580195)

[1.4 Welche Targets gibt es auf deinem System? Was bedeuten Sie? 2](#_Toc36580196)

[1.5 Welches Target wird beim Booten verwendet? 3](#_Toc36580197)

[1.6 Wie kann man einen Service starten/Stoppen/restarten? Wie kann man sich den Status anzeigen lassen? 3](#_Toc36580198)

[1.7 Wie kann man selbst eine unit hinzufügen? Schau dir dazu den Aufbau eines unit-Files an. 4](#_Toc36580199)

[1.7.1 Wo sind Unit-Files beheimatet? 4](#_Toc36580200)

[1.7.2 Aufbau einer Unit-Datei 5](#_Toc36580201)

[1.8 Wie kann ich mir die units eines tragets anzeigen lassen? Wie kann ich ein unit zu einem target hinzufügen? Wie kann ich es wieder entfernen? 6](#_Toc36580202)

[1.8.1 Hinzufügen einer Unit-Datei zu einem Target 6](#_Toc36580203)

[1.8.2 Entfernen des Units eines Targets 7](#_Toc36580204)

[1.9 Welche zusätzlichen Aufgaben übernimmt Systemd (bzw. was könnte es noch übernehmen)? Was sind die großen Kritikpunkte? 7](#_Toc36580205)

# Init-Systeme Linux

## Welches System (systemd oder init.d) verwendet dein Linux?

Mein System (Debian 10 Buster) verwendet systemd, weil Debian seit Version 8 standardmäßig SystemD verwendet.

root@nhaiden:~# dmesg | grep "systemd"

[37148.491144] systemd[1]: Inserted module 'autofs4'

[37148.556609] systemd[1]: systemd 241 running in system mode. (+PAM +AUDIT +SELINUX +IMA +APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD -IDN2 +IDN -PCRE2 default-hierarchy=hybrid)

## Gibt es im Verzeichnis /etc/init.d Dateien? Wozu?

root@nhaiden:~# ls /etc/init.d/

apache2 cloud-final docker ntp resolvconf sudo webmin

apache-htcacheclean cloud-init hwclock.sh openvpn rsync udev x11-common

apparmor cloud-init-local kmod pcscd rsyslog unattended-upgrades

cgroupfs-mount cron mysql procps screen-cleanup unscd

cloud-config dbus networking qemu-guest-agent ssh vsftpd

In diesem Ordner befinden sich Dateien von Diensten, die beim Systemstart automatisch mitgestartet werden sollen.

## Welche units gibt es auf deinem System?

Um sich die Units anzeigen zu lassen, kann man einfach systemctl list-units eingeben. Dieser Command listet alle Units auf, zu denen auch Devices gehören. Um sich nur die Services anzeigen zu lassen, kann man den Parameter –type=service hinzufügen.

root@nhaiden:~# systemctl list-units --type=service

UNIT LOAD ACTIVE SUB DESCRIPTION

apache2.service loaded active running The Apache HTTP Server

apparmor.service loaded active exited Load AppArmor profiles

cloud-config.service loaded active exited Apply the settings specified in cl

cloud-final.service loaded active exited Execute cloud user/final scripts

cloud-init-local.service loaded active exited Initial cloud-init job (pre-networ

cloud-init.service loaded active exited Initial cloud-init job (metadata s

containerd.service loaded active running containerd container runtime

cron.service loaded active running Regular background program process

dbus.service loaded active running D-Bus System Message Bus

docker.service loaded active running Docker Application Container Engin

getty@tty1.service loaded active running Getty on tty1

ifupdown-pre.service loaded active exited Helper to synchronize boot up for

iptables-openvpn.service loaded active exited iptables rules for OpenVPN

kmod-static-nodes.service loaded active exited Create list of required static dev

## Welche Targets gibt es auf deinem System? Was bedeuten Sie?

root@nhaiden:~# systemctl list-units --type target  
UNIT LOAD ACTIVE SUB DESCRIPTION  
basic.target loaded active active Basic System  
cloud-config.target loaded active active Cloud-config availability  
cloud-init.target loaded active active Cloud-init target  
cryptsetup.target loaded active active Local Encrypted Volumes  
getty.target loaded active active Login Prompts  
graphical.target loaded active active Graphical Interface  
local-fs-pre.target loaded active active Local File Systems (Pre)  
local-fs.target loaded active active Local File Systems  
multi-user.target loaded active active Multi-User System  
network-online.target loaded active active Network is Online  
network-pre.target loaded active active Network (Pre)  
network.target loaded active active Network  
paths.target loaded active active Paths  
remote-fs.target loaded active active Remote File Systems  
slices.target loaded active active Slices  
sockets.target loaded active active Sockets  
swap.target loaded active active Swap  
sysinit.target loaded active active System Initialization  
time-sync.target loaded active active System Time Synchronized  
timers.target loaded active active Timers

Targets sind eine moderne Variante der SysVinit (des sehr alten Initd-Systems) Runlevels. Runlevels sind ein Levelsystem, bei dem der aktuelle Status eines Systems festgelegt wird (z.B neustarten). Hierbei gibt es verschiedene Targets, die von SystemD vorgegeben werden. Targets werden nach einem Gewissen Nutzen definiert und stellen sicher dass die Abhängigkeiten, die diese Targets voraussetzen, erfüllt werden. Ein normales, mit standardmäßigen Einstellungen konfiguriertes System bootet in das default.target, welches im Ordner /lib/system/system liegt. Targets sind hierbei Dateien mit einer gewissen Konfiguration.

Hierbei gibt es viele andere Targets, die von Programmen definiert werden können (siehe obige Ausgabe, z.B. cloud-init.target).

## Welches Target wird beim Booten verwendet?

Bei einer normal konfigurierten Distribution mit SystemD Init System wird das default.target verwendet.

## Wie kann man einen Service starten/Stoppen/restarten? Wie kann man sich den Status anzeigen lassen?

Um sich Informationen zu einem Prozess anzeigen zu lassen, verwendet man den systemctl (System-Control) Command der vom SystemD Daemon bereitgestellt wird. Dieses Programm kann den Status der unter SystemD gestarteten Dienste auslesen, sie neustarten, stoppen oder starten. Diese Befehle müssen als Root-Benutzer ausgeführt werden (bzw. mit sudo davor).

Starten:

systemctl start <SERVICE>.service

Stoppen:

systemctl stop <SERVICE>.service

Neustarten:

systemctl restart <SERVICE>.service

Den aktuellen Status des Dienstes anzeigen lassen:

systemctl status <SERVICE>.service

Beispielhaft wird dies hier am Apache2-Webserver demonstriert:

Status anzeigen:

root@nhaiden:~# systemctl status apache2  
● apache2.service - The Apache HTTP Server  
 Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)  
 Active: active (running) since Mon 2020-03-09 08:44:10 UTC; 3 weeks 0 days ago  
 Docs: https://httpd.apache.org/docs/2.4/  
 Process: 510 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)  
 Process: 1762 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)  
 Main PID: 689 (apache2)  
 Tasks: 7 (limit: 2374)  
 Memory: 345.0M  
 CGroup: /system.slice/apache2.service  
 ├─ 689 /usr/sbin/apache2 -k start  
 ├─1821 /usr/sbin/apache2 -k start  
 ├─1822 /usr/sbin/apache2 -k start  
 ├─1823 /usr/sbin/apache2 -k start  
 ├─1824 /usr/sbin/apache2 -k start  
 ├─1825 /usr/sbin/apache2 -k start  
 └─2668 /usr/sbin/apache2 -k start  
  
Mar 31 00:00:00 nhaiden.tech systemd[1]: Reloading The Apache HTTP Server.  
Mar 31 00:00:02 nhaiden.tech systemd[1]: Reloaded The Apache HTTP Server.

Neustarten des Dienstes:

root@nhaiden:~# systemctl restart apache2  
root@nhaiden:~#

Die extra Linie der Konsole wurde hier eingefügt um deutlich zu machen, dass der Systemctl-Befehl nur beim Anzeigen des Status eine Ausgabe erzeugt, beim Starten usw. allerdings nur wenn ein Startfehler auftritt.

Starten/Stoppen:

root@nhaiden:~# systemctl stop apache2  
root@nhaiden:~# systemctl start apache2  
root@nhaiden:~#

## Wie kann man selbst eine unit hinzufügen? Schau dir dazu den Aufbau eines unit-Files an.

### Wo sind Unit-Files beheimatet?

Unit Files finden sich in 3 Ordnern auf der Root-Festplatte wieder. Diese werden nach einer hard-codierten Reihenfolge (abhängig von Distribution und deren SystemD-Implementierung) abgearbeitet.

Eine typische Reihenfolge könnte so aussehen:

/etc/systemd/system # Lokale Konfiguration

/run/systemd/system # Dyanmisch erzeugte Unit-Dateien

/lib/systemd/system # Unit = Dateien für Distributionspakete

Befinden sich nun z.B. zwei Dateien desselben Namens, eine im Ordner /etc/… und eine in /run/… so wird diese, welche sich in etc befindet, von SystemD eingelesen und diese, die sich in run befindet, ignoriert und nicht eingelesen.

### Aufbau einer Unit-Datei

Schauen wir uns den Aufbau beispielhaft an der Unit-Datei von dem OpenSSH-Server Dienst an:

root@nhaiden:/etc/systemd/system# cat sshd.service  
[Unit]  
Description=OpenBSD Secure Shell server  
Documentation=man:sshd(8) man:sshd\_config(5)  
After=network.target auditd.service  
ConditionPathExists=!/etc/ssh/sshd\_not\_to\_be\_run  
  
[Service]  
EnvironmentFile=-/etc/default/ssh  
ExecStartPre=/usr/sbin/sshd -t  
ExecStart=/usr/sbin/sshd -D $SSHD\_OPTS  
ExecReload=/usr/sbin/sshd -t  
ExecReload=/bin/kill -HUP $MAINPID  
KillMode=process  
Restart=on-failure  
RestartPreventExitStatus=255  
Type=notify  
RuntimeDirectory=sshd  
RuntimeDirectoryMode=0755  
  
[Install]  
WantedBy=multi-user.target  
Alias=sshd.service

Eine Unit-Datei ist in 3 Abschnitte unterteilt:

1. [Unit], der oberste Teil der Unit Datei, enthält generelle Informationen über einen Service. Hier stehen Dinge wie eine Beschreibung (Description) über den Service, Links / Verweise auf Dokumentationen (Documentation). After ist hier eine Option, die dazu dient festzulegen, nach welchen Diensten der SSH-Dienst vom systemD Daemon gestartet werden soll. So ist es nicht sinnvoll, den SSH Dienst vor dem Netzwerk Dienst zu starten, wenn das System noch keine IP-Adresse hat.
2. [Service] beschreibt generelle Informationen zu einem Dienst, wie er sich in gewissen Situationen zu verhalten hat (reload, restart usw.). So definieren die ExecReload, ExecStart Parameter, was der Dienst bei einem Reload bzw. Neustart zu tun hat bzw. tun sollte.

Der KillMode legt den Modus fest, in welchen der Dienst gekillt werden soll.

1. [Install] beschreibt Informationen die für das Installieren, das Starten einer Unit bzw. Dienst notwendig sind. Hierbei kann man angeben, welche Abhängigkeiten an Targets bzw. Diensten vorliegen müssen, damit dieser Service startet bzw. gestartet werden kann.

Alias kann man verwenden, um einem Dienst, der vielleicht einen längeren Namen hat, einen kurzen zu geben, damit man beim Administrieren (Neustarten usw.) nicht viel in die Shell eingeben muss.

## Wie kann ich mir die units eines tragets anzeigen lassen? Wie kann ich ein unit zu einem target hinzufügen? Wie kann ich es wieder entfernen?

Anzeigen lassen der Units eines Targets

root@nhaiden:/etc/systemd/system# systemctl list-dependencies cloud-init.target  
cloud-init.target  
● ├─cloud-config.service  
● ├─cloud-final.service  
● ├─cloud-init-local.service  
● └─cloud-init.service

### Hinzufügen einer Unit-Datei zu einem Target

Machen wir dies am Beispiel einer einfachen Unit-Datei:

root@nhaiden:/etc/systemd/system# cat testservice.service  
[Unit]  
Description = NVS Unit  
After = network.target  
  
[Service]  
ExecStart = /root/scripts/test.sh  
  
  
[Install]  
WantedBy = multi-user.target

Wichtig hier ist das WantedBy in dem Install-Bereich der Unit-Datei. Dies legt fest, zu welchem Target die Unit-Datei gehören soll.

In diesem Fall soll dieser Dienst nach dem Networking-Dienst gestartet werden. Wenn er startet, führt er das Script test.sh aus. Der Start passiert bei Booten des Multi-User-Modus, also den ganz normalen Start des Linux-Systems.

Um diesen Service nun zum multi-user.target hinzuzufügen, gibt man einfach

systemctl enable testservice.service

Ein. Der Systemctl Befehl erstellt automatisch einen symantischen Link in das Verzeichnis des multi-user.target’s.

root@nhaiden:/etc/systemd/system# systemctl enable testservice.ser  
vice  
Created symlink /etc/systemd/system/multi-user.target.wants/testse  
rvice.service → /etc/systemd/system/testservice.service.

root@nhaiden:/etc/systemd/system# ls multi-user.target.wants/  
apache2.service openvpn.service  
containerd.service remote-fs.target  
cron.service rsync.service  
docker.service rsyslog.service  
iptables-openvpn.service ssh.service  
mariadb.service testservice.service

### Entfernen des Units eines Targets

Nehmen wir das Beispiel von vorher her und entfernen des vom Multi-User Target.

Dies geschieht über ein Einfaches

systemctl disable testservice.service

Der Systemctl-Befehl entfernt automatisch den Testservice-Link aus dem Verzeichnis und somit wird dieser nicht mehr beim ausführen des Targets gestartet.

root@nhaiden:/etc/systemd/system# systemctl disable testservice.se  
rvice  
Removed /etc/systemd/system/multi-user.target.wants/testservice.se  
rvice.

## Welche zusätzlichen Aufgaben übernimmt Systemd (bzw. was könnte es noch übernehmen)? Was sind die großen Kritikpunkte?

SystemD polarisiert in der Linux-Community sehr stark und ist ein sehr umstrittenes Init-System.

Ein besonders kritisiertes Merkmal von SystemD ist, dass Logs nicht in normalen Textdateien, sondern in binärem Format abgelegt werden, sodass man ein extra Programm braucht, um diese auszulesen und kann nicht wie bei SysVInit oder anderen Init-Systemen einfache Programme wie cat usw. benutzen.

Ein weiteres massiver Kritikpunkt, der oft an SystemD geäußert wird, sind dass viele kleine Dienste, die von den SystemD Entwicklern SystemD als riesige Dependency brauchen. Viele sagen auch, dass manche Programme, die kein Init-System als Dependency bräuchten, SystemD als Dependency drin haben.